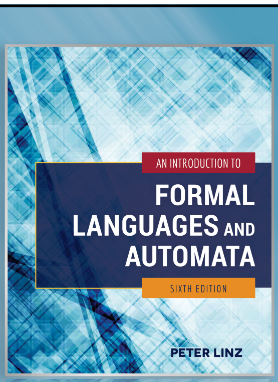
Chapter 7

PUSHDOWN AUTOMATA



Learning Objectives

At the conclusion of the chapter, the student will be able to:

- Describe the components of a nondeterministic pushdown automaton
- State whether an input string is accepted by a nondeterministic pushdown automaton
- Construct a pushdown automaton to accept a specific language
- Given a context-free grammar in Greibach normal form, construct the corresponding pushdown automaton
- Describe the differences between deterministic and nondeterministic pushdown automata
- Describe the differences between deterministic and general context-free languages

---

Nondeterministic Pushdown Automata

- A pushdown automaton is a model of computation designed to process context-free languages
- Pushdown automata use a stack as storage mechanism

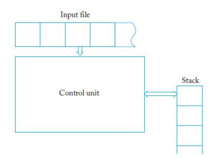


FIGURE 7.1

Nondeterministic Pushdown Automata

- A *nondeterministic pushdown accepter* (npda) is defined by:
  - A finite set of states Q
  - An input alphabet Σ
  - A stack alphabet Γ
  - A transition function δ
  - An initial state q_0
  - A stack start symbol z
  - A set of final states F
- Input to the transition function δ consists of a triple consisting of a state, input symbol (or λ), and the symbol at the top of stack
- Output of δ consists of a new state and new top of stack
- Transitions can be used to model common stack operations

Sample npda Transitions

- Example 7.1 presents the sample transition rule:

  $\delta(q_1, a, b) = \{(q_2, cd), (q_3, \lambda)\}$

- According to this rule, when the control unit is in state q_1, the input symbol is a, and the top of the stack is b, two moves are possible:
  - New state is q_2 and the symbols cd replace b on the stack
  - New state is q_3 and b is simply removed from the stack
- If a particular transition is not defined, the corresponding (state, symbol, stack top) configuration represents a *dead* state

A Sample Nondeterministic Pushdown Accepter

- Example 7.2: Consider the npda

  $Q = \{ q_0, q_1, q_2, q_3 \}$, $\Sigma = \{ a, b \}$, $\Gamma = \{ 0, 1 \}$, $z = 0$, $F = \{q_3\}$

  with initial state q_0 and transition function given by:

  $\delta(q_0, a, 0) = \{ (q_1, 10), (q_3, \lambda) \}$
  $\delta(q_0, \lambda, 0) = \{ (q_3, \lambda) \}$
  $\delta(q_1, a, 1) = \{ (q_1, 11) \}$
  $\delta(q_1, b, 1) = \{ (q_2, \lambda) \}$
  $\delta(q_2, b, 1) = \{ (q_2, \lambda) \}$
  $\delta(q_2, \lambda, 0) = \{ (q_3, \lambda) \}$
- As long as the control unit is in q_1, a 1 is pushed onto the stack when an a is read
- The first b causes control to shift to q_2, which removes a symbol from the stack whenever a b is read

Transition Graphs

- In the transition graph for a npda, each edge is labeled with the input symbol, the stack top, and the string that replaces the top of the stack
- The graph below represents the npda in Example 7.2:

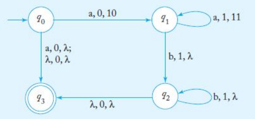
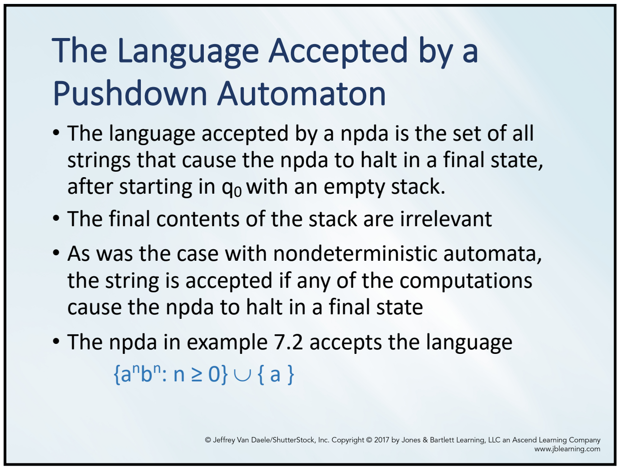


FIGURE 7.2

Instantaneous Descriptions

- To trace the operation of a npda, we must keep track of the current state of the control unit, the stack contents, and the unread part of the input string
- An *instantaneous description* is a triplet (q, w, u) that describes state, unread input symbols, and stack contents (with the top as the leftmost symbol)
- A move is denoted by the symbol $\vdash$
- A partial trace of the npda in Example 7.2 with input string ab is

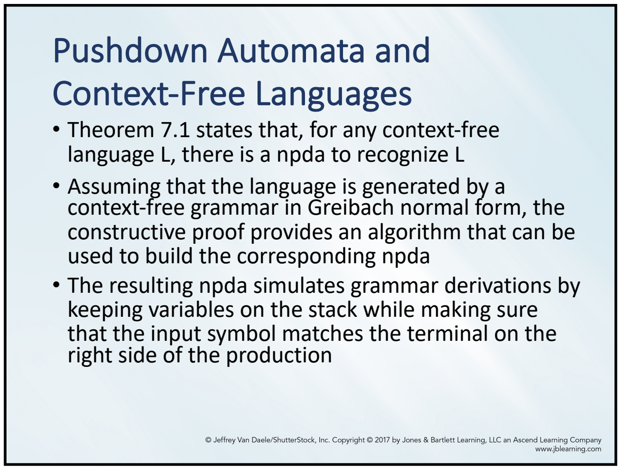
  $(q_0, ab, 0) \vdash (q_1, b, 10) \vdash (q_2, \lambda, 0) \vdash (q_3, \lambda, \lambda)$



The Language Accepted by a Pushdown Automaton

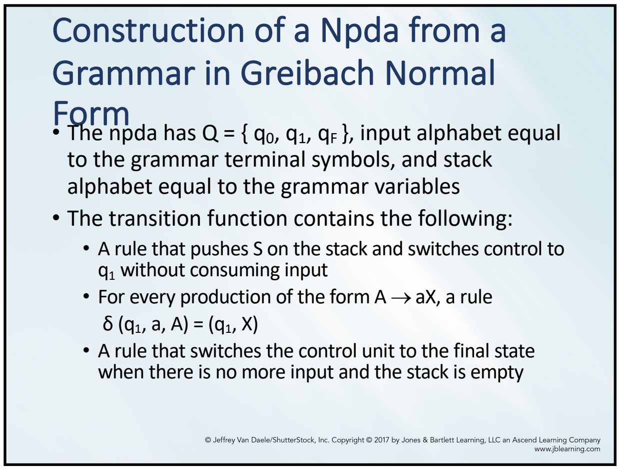
- The language accepted by a npda is the set of all strings that cause the npda to halt in a final state, after starting in q_0 with an empty stack.
- The final contents of the stack are irrelevant
- As was the case with nondeterministic automata, the string is accepted if any of the computations cause the npda to halt in a final state
- The npda in example 7.2 accepts the language

  $\{a^n b^n : n \geq 0\} \cup \{ a \}$



Pushdown Automata and Context-Free Languages

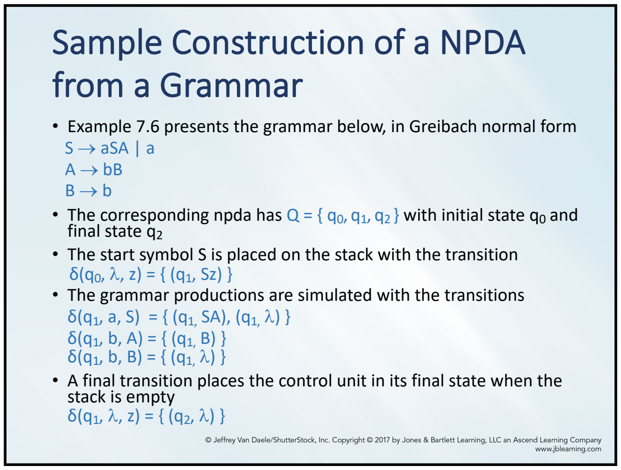
- Theorem 7.1 states that, for any context-free language L, there is a npda to recognize L
- Assuming that the language is generated by a context-free grammar in Greibach normal form, the constructive proof provides an algorithm that can be used to build the corresponding npda
- The resulting npda simulates grammar derivations by keeping variables on the stack while making sure that the input symbol matches the terminal on the right side of the production



Construction of a Npda from a Grammar in Greibach Normal Form

- The npda has $Q = \{ q_0, q_1, q_F \}$, input alphabet equal to the grammar terminal symbols, and stack alphabet equal to the grammar variables
- The transition function contains the following:
  - A rule that pushes S on the stack and switches control to q_1 without consuming input
  - For every production of the form $A \rightarrow aX$, a rule

    $\delta (q_1, a, A) = (q_1, X)$
  - A rule that switches the control unit to the final state when there is no more input and the stack is empty



Sample Construction of a NPDA from a Grammar

- Example 7.6 presents the grammar below, in Greibach normal form

  $S \rightarrow aSA \mid a$

  $A \rightarrow bB$

  $B \rightarrow b$
- The corresponding npda has $Q = \{ q_0, q_1, q_2 \}$ with initial state q_0 and final state q_2
- The start symbol S is placed on the stack with the transition

  $\delta(q_0, \lambda, z) = \{ (q_1, Sz) \}$
- The grammar productions are simulated with the transitions

  $\delta(q_1, a, S) = \{ (q_1, SA), (q_1, \lambda) \}$

  $\delta(q_1, b, A) = \{ (q_1, B) \}$

  $\delta(q_1, b, B) = \{ (q_1, \lambda) \}$
- A final transition places the control unit in its final state when the stack is empty

  $\delta(q_1, \lambda, z) = \{ (q_2, \lambda) \}$

Deterministic Pushdown Automata

- A *deterministic pushdown accepter* (dpda) never has a choice in its move
- Restrictions on dpda transitions:
  - Any (state, symbol, stack top) configuration may have at most one (state, stack top) transition definition
  - If the dpda defines a transition for a particular (state, λ, stack top) configuration, there can be no input-consuming transitions out of state s with a at the top of the stack
- Unlike the case for finite automata, a λ-transition does not necessarily mean the automaton is nondeterministic

Example of a Deterministic Pushdown Automaton

- Example 7.10 presents a dpda to accept the language

  $L = \{ a^n b^n : n \geq 0 \}$
- The dpda has $Q = \{ q_0, q_1, q_2 \}$, input alphabet { a, b }, stack alphabet { 0, 1 }, z = 0, and q_0 as its initial and final state
- The transition rules are

  $\delta(q_0, a, 0) = \{ (q_1, 10) \}$

  $\delta(q_1, a, 1) = \{ (q_1, 11) \}$

  $\delta(q_1, b, 1) = \{ (q_2, \lambda) \}$

  $\delta(q_2, b, 1) = \{ (q_2, \lambda) \}$

  $\delta(q_2, \lambda, 0) = \{ (q_0, \lambda) \}$

Deterministic Context-Free Languages

- A context-free language L is *deterministic* if there is a dpda to accept L
- Sample deterministic context-free languages:

  $\{ a^n b^n : n \geq 0 \}$

  $\{ wxw^R : w \in \{a, b\}^* \}$
- Deterministic and nondeterministic pushdown automata are not equivalent: there are some context-free languages for which no dpda can be built